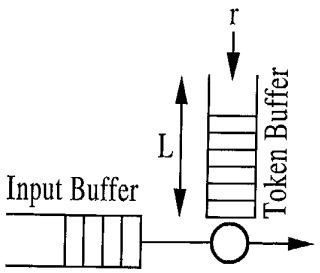

FIG. 1

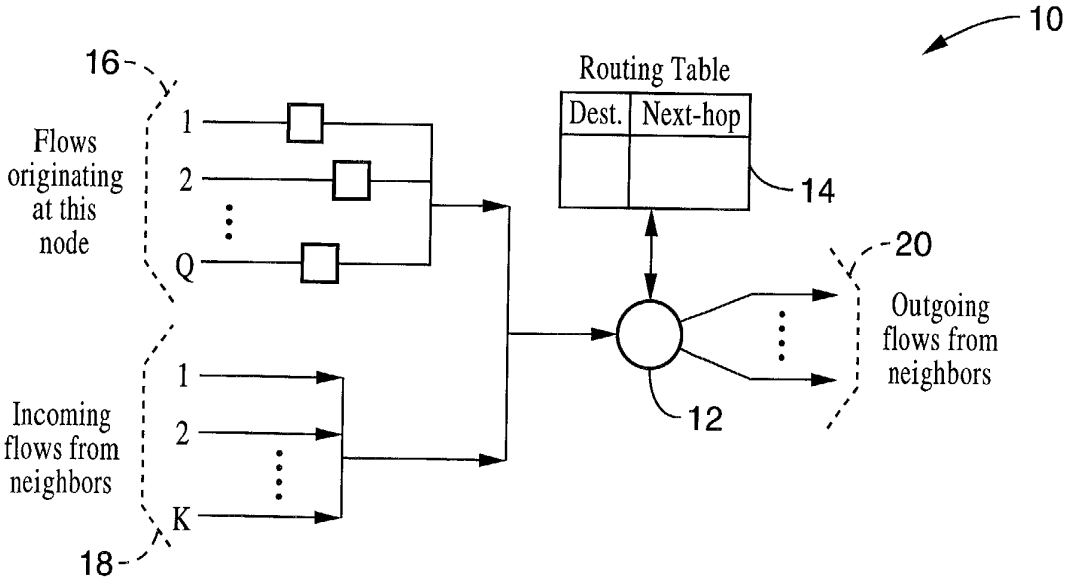

FIG. 2

30

```
01  PROCEDURE AGRA()

02   if a REFRESH message arrived from k with bandwidth b for j.
03       B^i_jk ← b;
04  if a timeout occured, then for each destination j,
05      let the refresh mesages for j recieved during the
06          last refresh period T_R have a total bandwidth of BT^i_j;
07      if node is in READY state for j and BT^i_j > B^i_js
08          CALL DIFFCOMP (BT^i_j - B^i_js);
09      B^i_js ← MIN {B^i_js, BT^i_j};
10      send REFRESH message to s with bandwidth B^i_j;
11  if link to k failed, for all j set B^i_jk ← 0.
12  if a RELEASE message is received from k with bandwidth b for j,
13          B^i_js ← B^i_js - b;
14          if node is READY for j, CALL DIFFCOMP (b);
15          otherwise the node is in WAIT state for j, send [ACK,j] to k;
16  if last ACK message is received for j,
17      become READY for j,
18      send [ACK, j] to s;
```

40

```
    PROCEDURE DIFFCOMP(j, b)

01  distribute b among this node and all predecessor neighbors;
02  terminate as many flows at this node to satisfy this node's share;
03  for each predecessor neighbor k and its share of b_k,
04      send [RELEASE, j, b_k] to k;
05  set node as WAIT for j;
```

FIG. 3

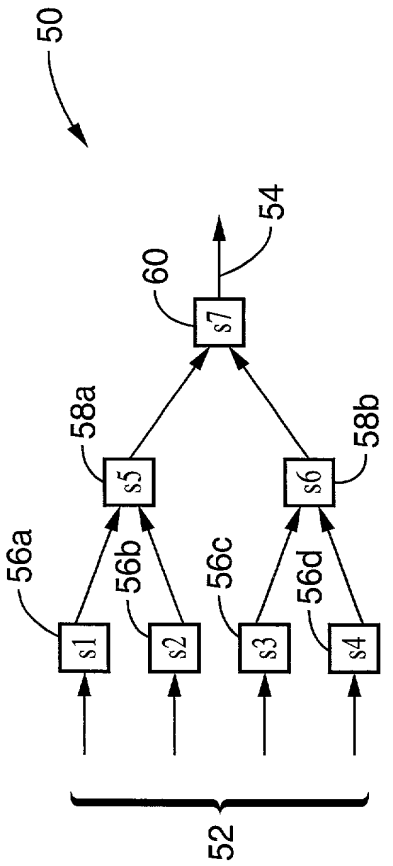
50
54
60
s7
58a
58b
s5
s6
56a
56b
56c
56d
s1
s2
s3
s4
52

FIG. 4

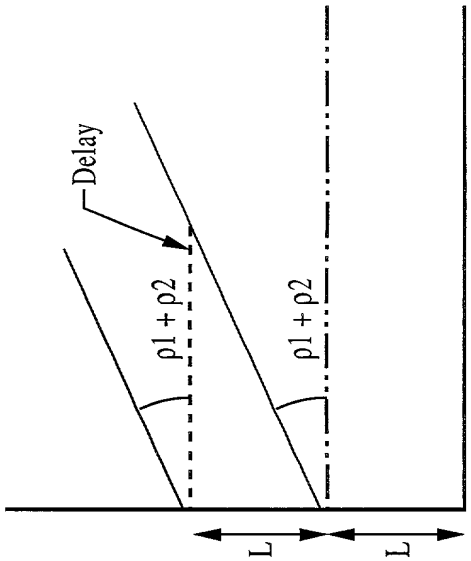
Delay
ρ1 + ρ2
ρ1 + ρ2
L
L

FIG. 5

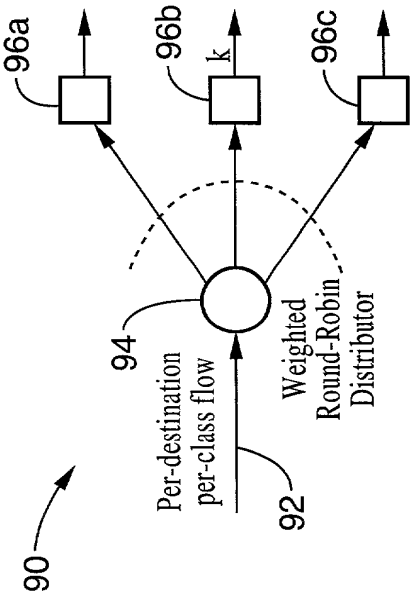
96a
96b
k
96c
94
Per-destination
per-class flow
Weighted
Round-Robin
Distributor
92
90

FIG. 7

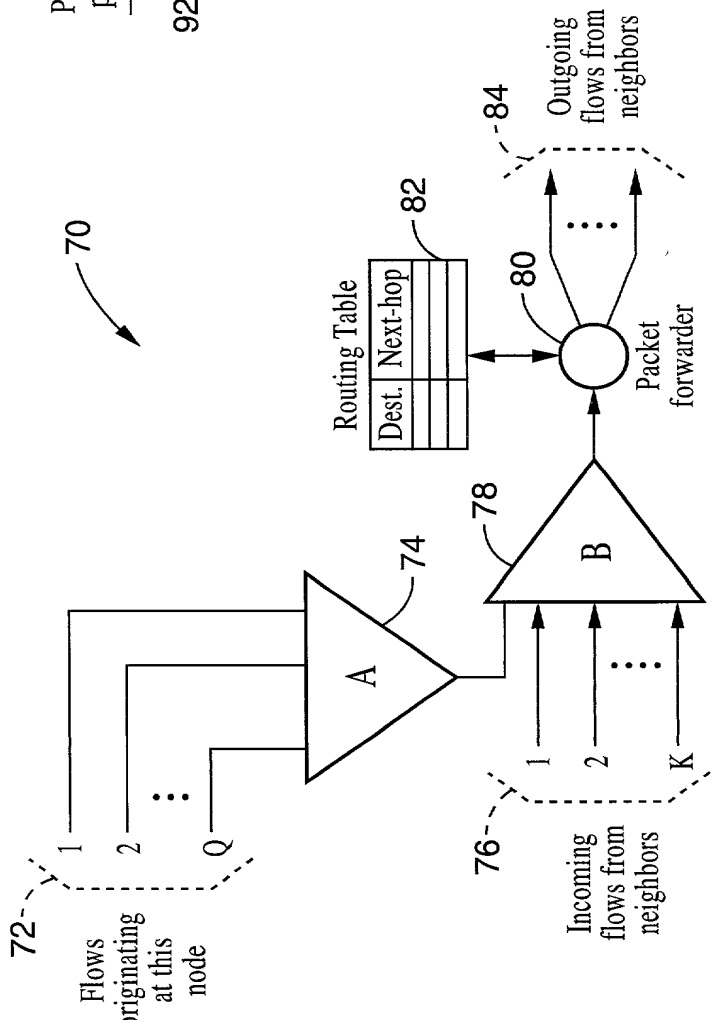
70
Routing Table
Dest.
Next-hop
82
80
84
Outgoing
flows from
neighbors
Packet
forwarder
78
B
74
A
1
2
Q
72
Flows
originating
at this
node
1
2
K
76
Incoming
flows from
neighbors

FIG. 6

100

```
01  PROCEDURE Multipath-packet-Forwarding(P)
02  {Executed at router i on arrival of a packet for j.}
03  BEGIN
04      φ^i_jk ← B^i_jk / B^i_j for some k ∈ S^i_j;
05      Let W^i_jk ≤ φ^i_jk W^i_j for some k ∈ S^i_j;
06      Forward packet to neighbor k;
07      W^i_jk ← W^i_jk + length(P);
08      W^i_j ← W^i_j + length(P);
09  END
```

FIG. 8

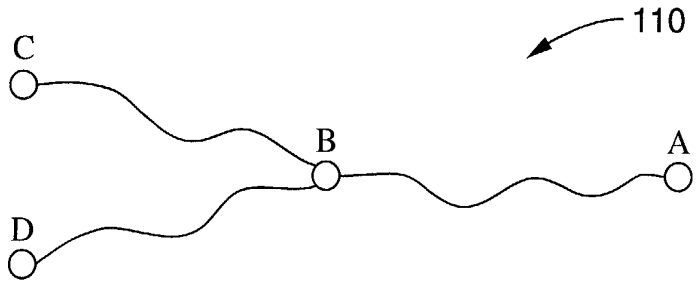

FIG. 9

120

```
01  PROCEDURE Refresh-PKT-LS-Arch(u)
02  {Periodically executed at node i, for label u}
03  BEGIN
04      Let the refresh messages from neighbors received in T_R specify
05          a total bandwidth of B^T_u for label u;
06      LET the r.t.e for u be (u, k, u, B_u);
07      B_u ← MIN (B_u, BT_u);
08      Send refresh message to k with bandwidth B_uj
09  END
```

FIG. 10

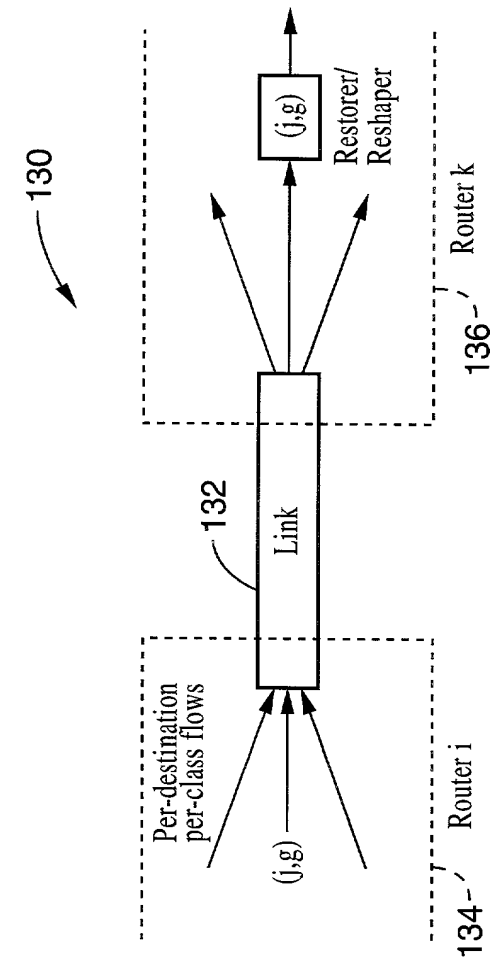
130
Per-destination per-class flows
(j,g)
132
Link
(j,g)
Restorer/ Reshaper
134
Router i
136
Router k

FIG. 11

150

```
01  PROCEDURE AgreeEvent (type, j, g, k, b)

02  if(type = REFRESH), B^i_{j,g,k} ← B^i_{j,g,k} + b;
03  if (type = TIMEOUT), then {
04      bt ← Σ_{k∉s_j^i} B^i_{j,g,k} + I^i_{j,g};
05      bw ← Σ_{k∉s_j^i} B^i_{j,g,k};
06      if (bt < bw), then {
07          Divide bt into bk such that Σb_k = bt and b_k ≤ B^i_{j,g,k};
08          B^i_{j,g,k} ← b_k;
09      }
10      if bt > bw and state^i_{j,g} = PASSIVE, then
11          CALL DiffComp (j, g, bt - bw);
12      for each k ∈ S^i_j, send [REFRESH, j, g, k, B^i_{j,k}];
13      }
14  if (type = RELEASE), then {
15          B^i_{j,g,k} ← B^i_{j,g,k} - b;
16          if (state^i_{j,g} = PASSIVE), then
17              CALL DiffComp (j, g, b);
            otherwise send [ACK, j, g] to k;
18  }
20  if (type = ACK and last ACK message for j and g) {
21          state^i_{j,g} ← PASSIVE
22          send [ACK, j, g] to s, if s is waiting for ACK;
    }
24  if(type = SETUP) then // s is the successor on the path
25          B^i_{j,g,k} ← B^i_{j,g,k} + b; B^i_{j,g,s} ← B^i_{j,g,s} + b;
26  if(type = TERMINATE) then
27          B^i_{j,g,k} ← B^i_{j,g,k} - b; B^i_{j,g,s} ← B^i_{j,g,s} - b;
```

160

```
28  PROCEDURE Diff Comp (j, g, b) at node i

29          if (b ≤ I^i_{j,g}), then terminate flows for j and
30              class g that add up to at least b and return;
32      br ← b - I^i_{j,g};
        Divide br into b_k and k ∉ S^i_j such that
33          Σb_k = br and b_k ≤ B^i_{j,g,k};
34      for each k ∉ S^i_j, send [RELEASE, j, g, b_k] to k;
35      state^i_{j,g} ← ACTIVE;
```

FIG. 12

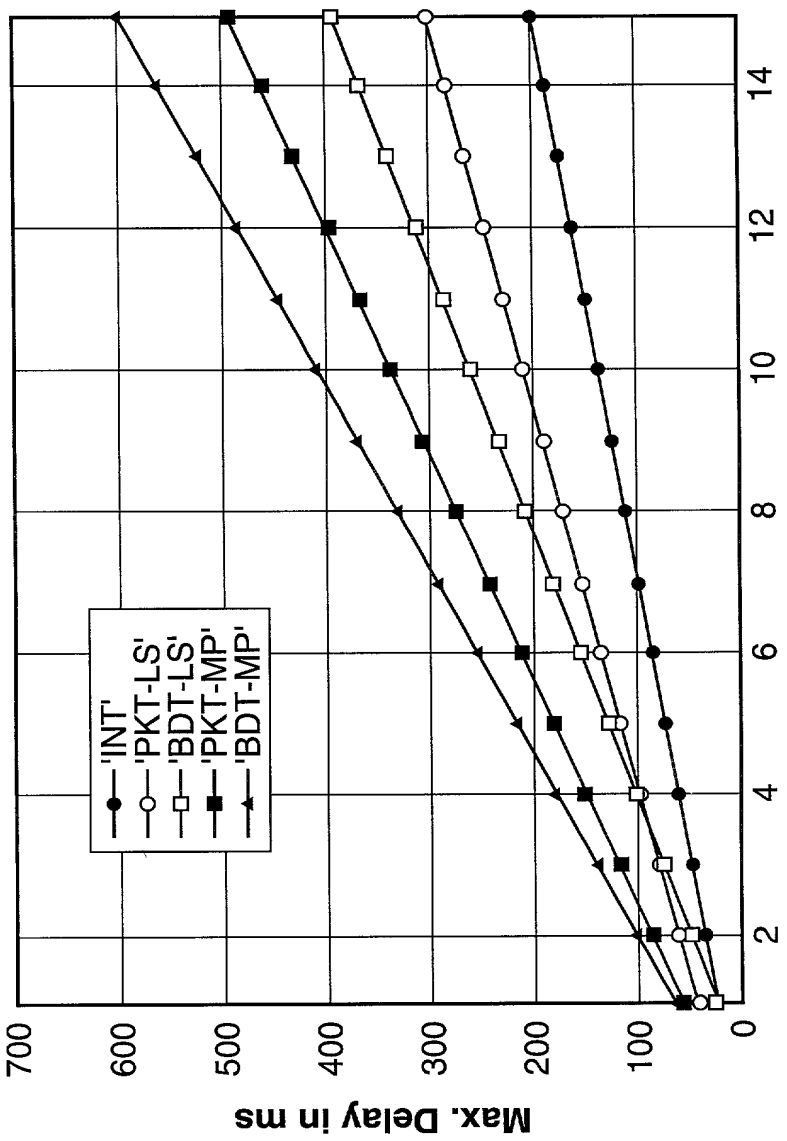
Max. Delay in ms
0
100
200
300
400
500
600
700
2
4
6
8
10
12
14
Path Length
'INT'
'PKT-LS'
'BDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 13

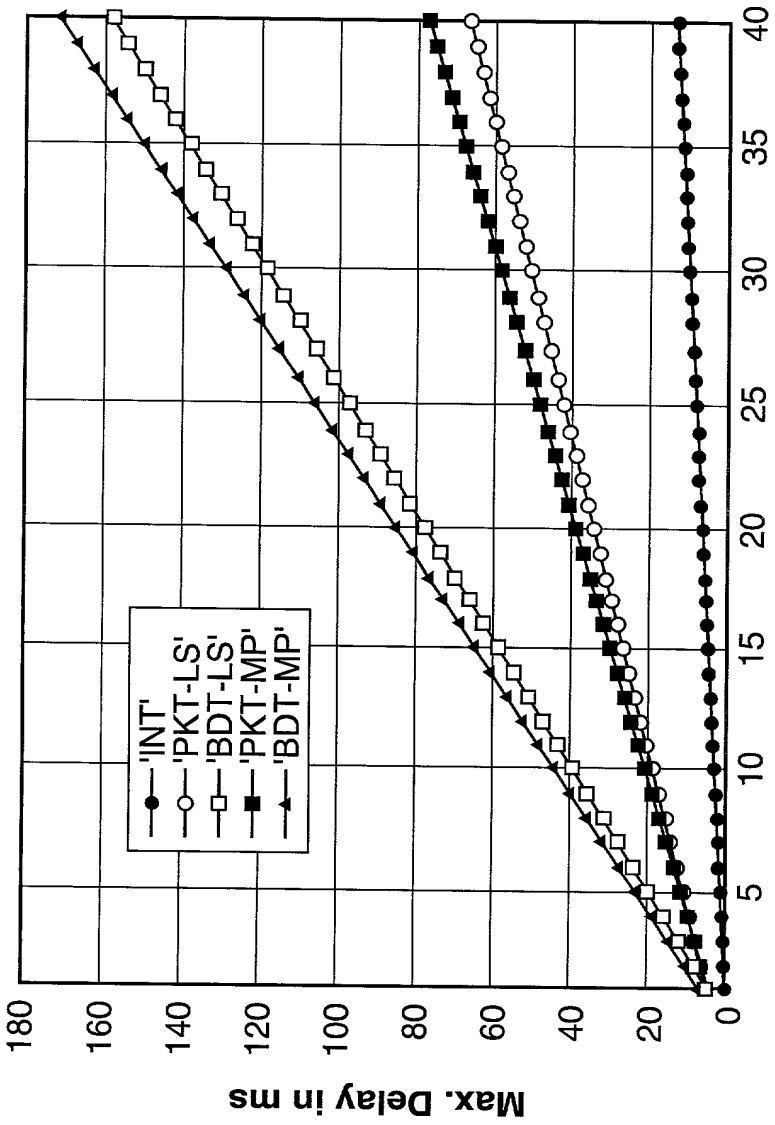
Max. Delay in ms
0
20
40
60
80
100
120
140
160
180
0
5
10
15
20
25
30
35
40
Path Length
'INT'
'PKT-LS'
'BDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 14

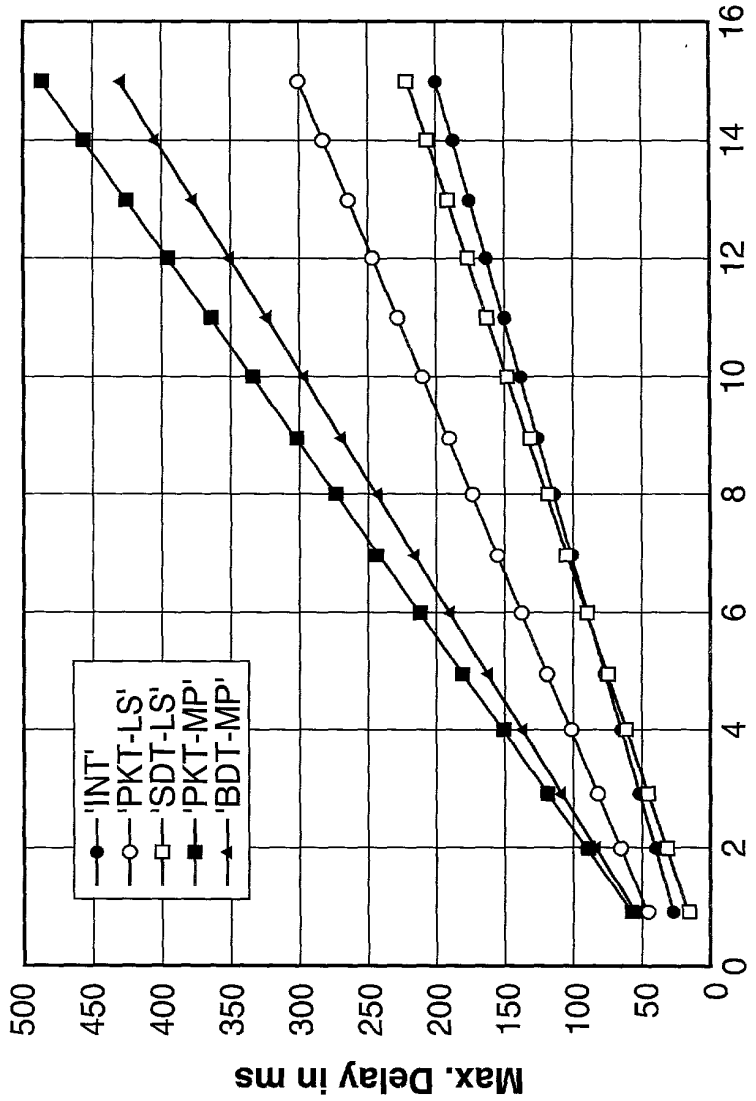
Max. Delay in ms
500
450
400
350
300
250
200
150
100
50
0
0
2
4
6
8
10
12
14
16
Path Length
'INT'
'PKT-LS'
'SDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 15

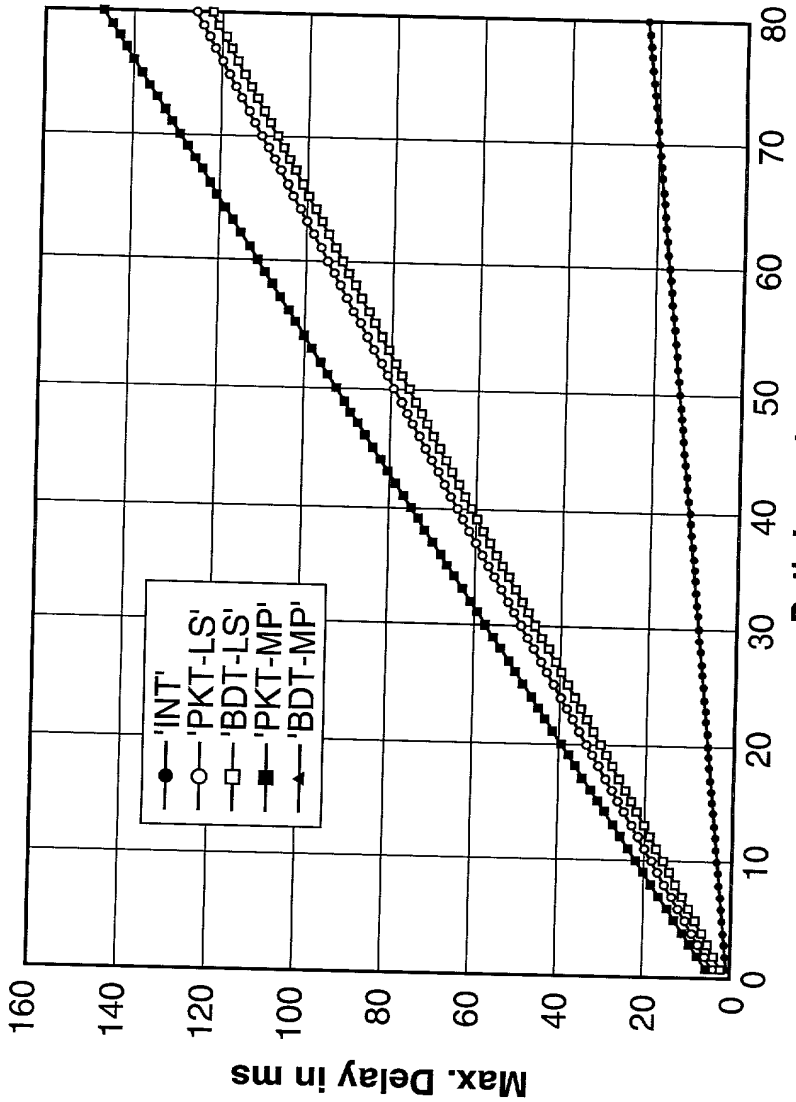
Max. Delay in ms
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
60
70
80
Path Length
'INT'
'PKT-LS'
'BDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 16

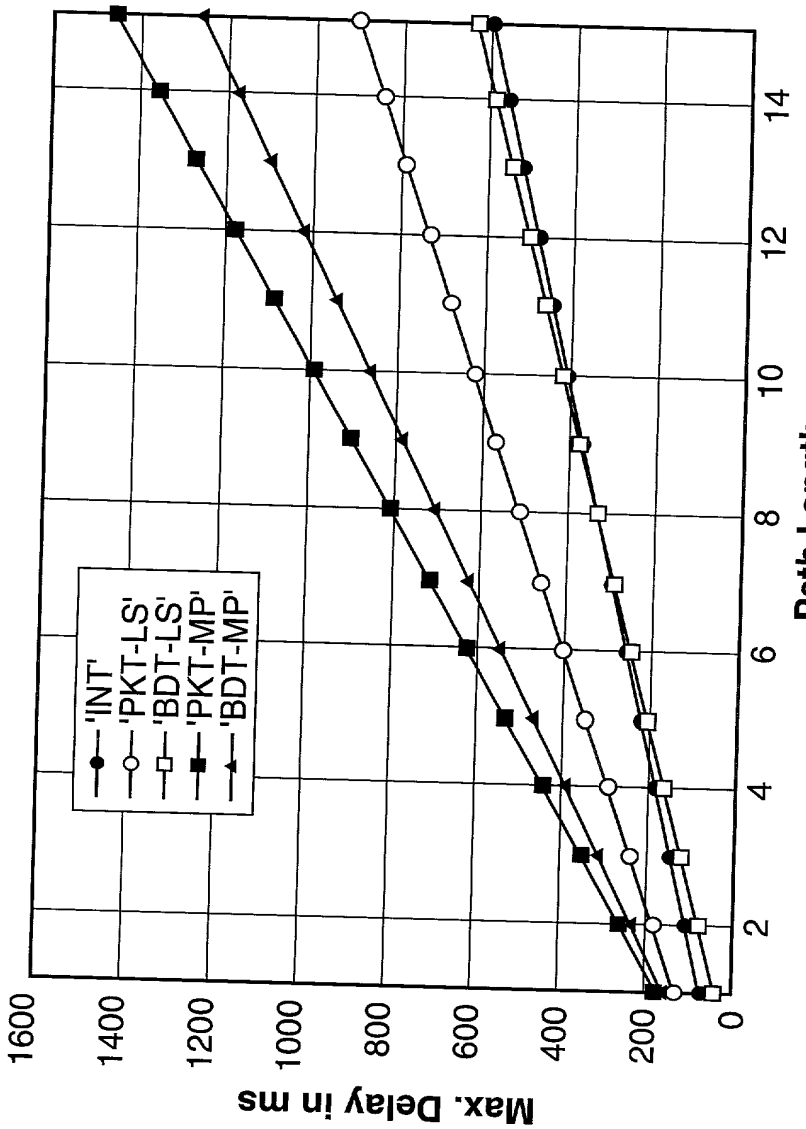
Max. Delay in ms
1600
1400
1200
1000
800
600
400
200
0
2
4
6
8
10
12
14
Path Length
'INT'
'PKT-LS'
'BDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 17

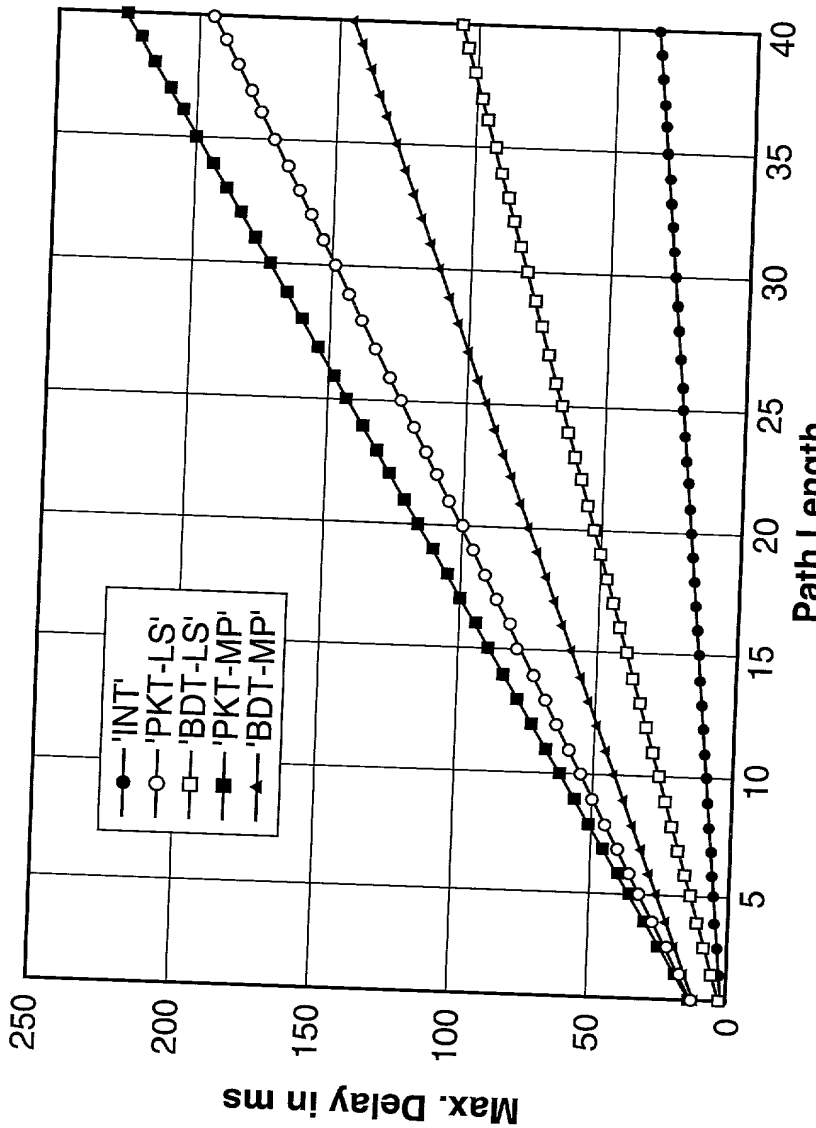
Max. Delay in ms
250
200
150
100
50
0
5
10
15
20
25
30
35
40
Path Length
'INT'
'PKT-LS'
'BDT-LS'
'PKT-MP'
'BDT-MP'

FIG. 18

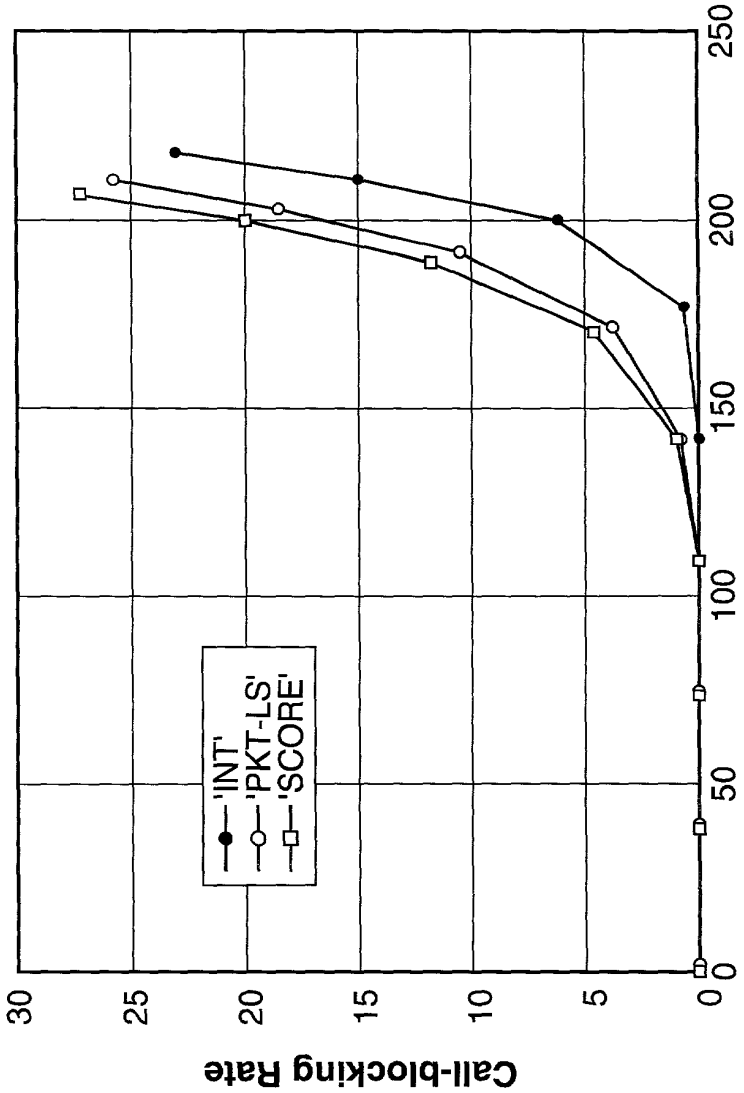
Call-blocking Rate
Load
'INT'
'PKT-LS'
'SCORE'
0
5
10
15
20
25
30
0
50
100
150
200
250

FIG. 19

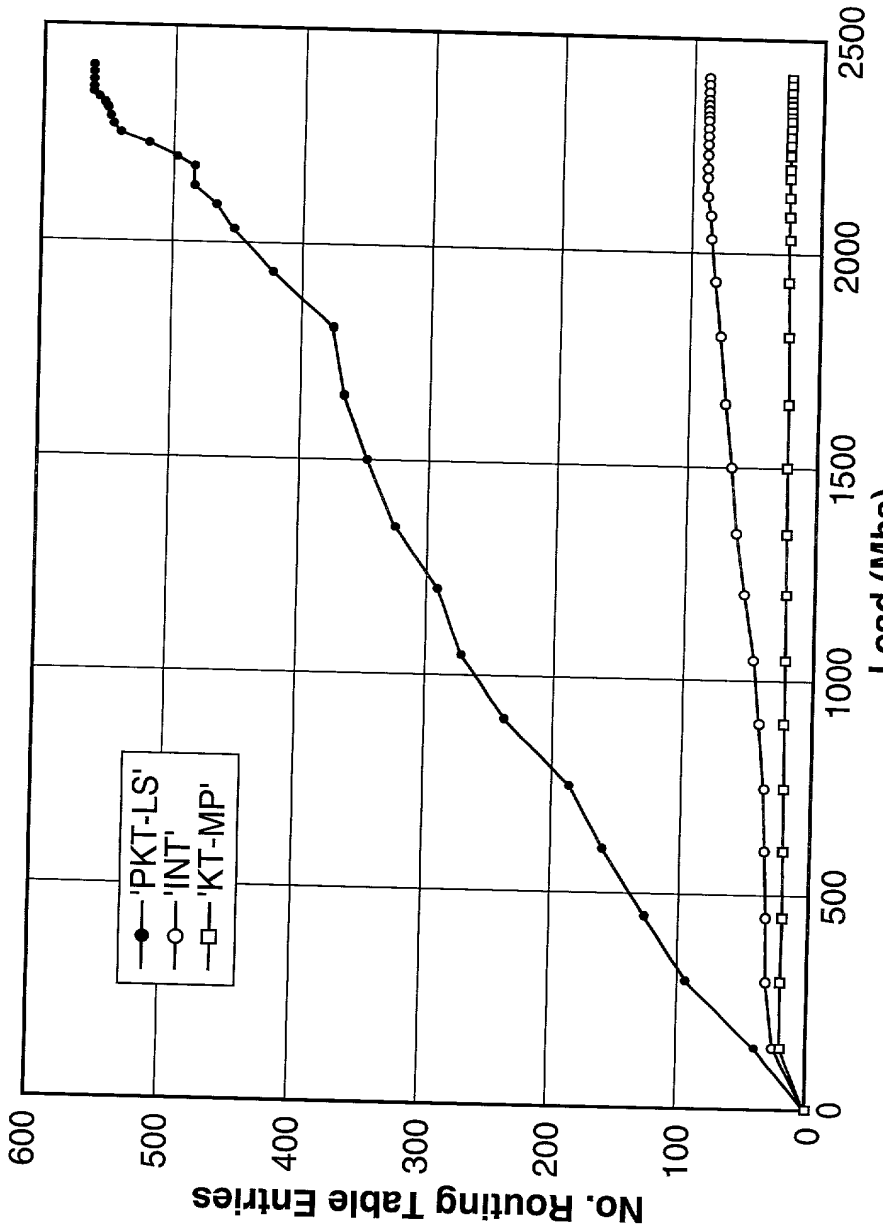
No. Routing Table Entries
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
Load (Mbs)
'PKT-LS'
'INT'
'KT-MP'

FIG. 20